

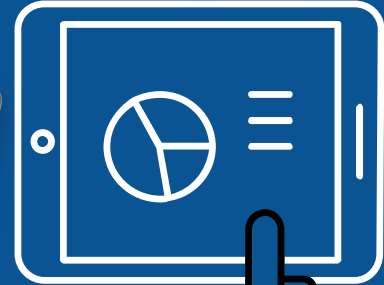
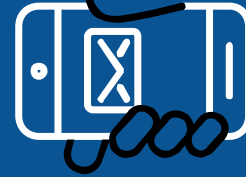
The background is a solid blue color. At the bottom, there is a stylized city skyline with various skyscrapers and buildings drawn in black outlines. Several white, fluffy clouds are scattered across the blue sky. In the center, there is a white rectangular box with a thick black border. Inside this box, the title is written in bold, black, italicized capital letters.

# ***AUTOMATIZACIÓN DE PRUEBAS CON SELENIUM Y APPIUM***



## Victor Portugal

Microsoft Certified Technology Specialist (MCTS).  
Certified Scrum Master (CSM).  
Certified Scrum Developer (CSD).  
Design Thinking Professional Certificate (DTPC)  
Associate Android Developer (AAD).





# **INTRODUCCIÓN**

- Introducción a la programación orientada a objetos.
- Creación de un proyecto java y primeras clases.
- Métodos.
- Estructuras de datos de java.
- Herramientas que vamos a usar
- Armado del proyecto de automation.
- Maven: armado del POM.xml



# ***SELENIUM***

- Instalando webdrivers
- Primeras pruebas usando Selenium
- Interactuando con el webdriver (click, sendKeys, etc.)
- Interactuando con un Select (combos)
- Nociones de Asserts.
- Waits (Implicit, Explicit, Fluent).
- Before y After (test, class, etc.)
- Page Objects.
- Separando casos (buenas prácticas)
- Page Factory
- Manipulando el browser
- Tomando screenshots
- Reportes
- Headless
- Parámetros del webdriver
- Describiendo resultados
- Test Suites
- Readers/Writers de archivos(data pool).
- Conexión a base de datos sql server.

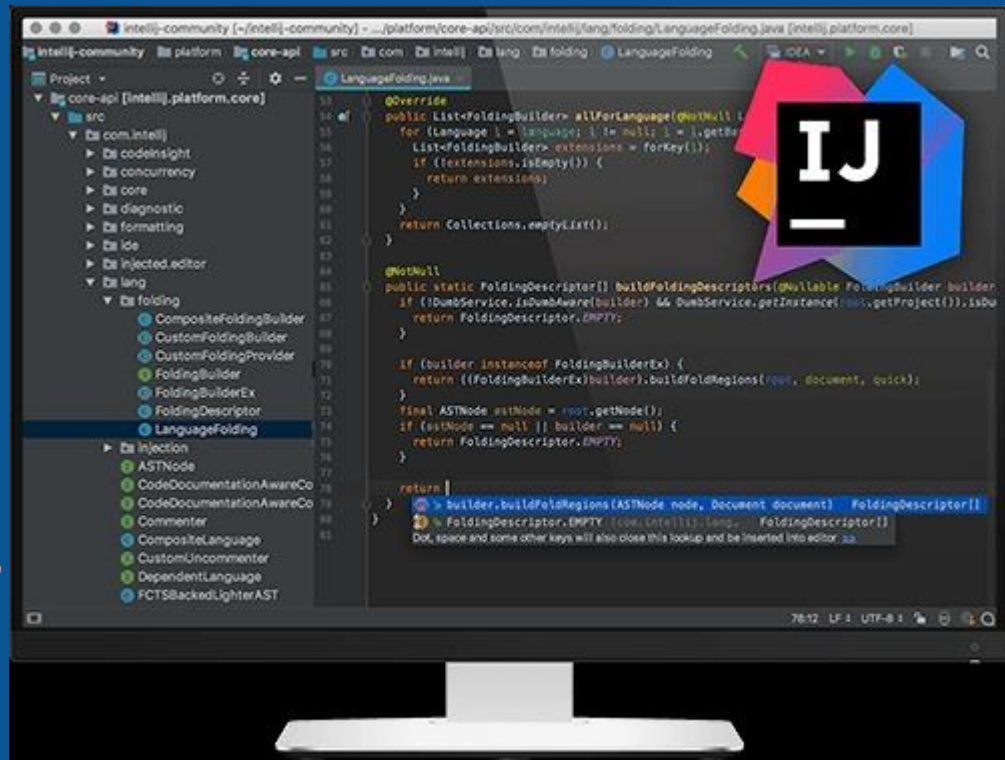
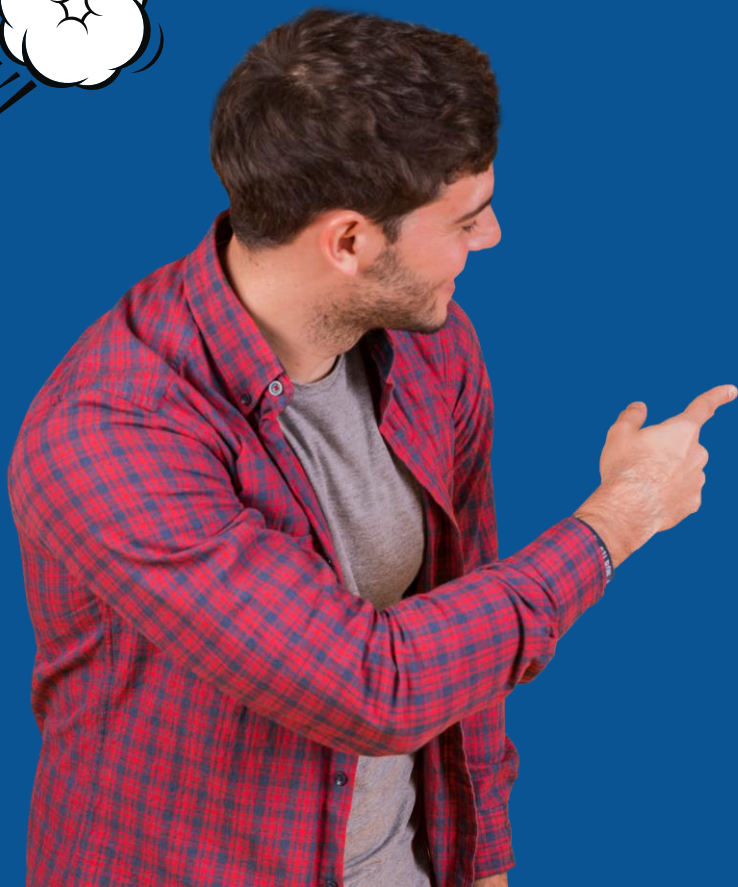


# ***APPIUM***

- Arquitectura
- Tipos de Pruebas Mobile
- Appium Inspector
- Interacciones en Appium
- Nuestro Primer Script con Appium
- Estrategias para pruebas con Appium
- Automatizando distintas apps
- Automatizando aplicaciones de escritorio
- Locators e Interacciones Avanzados
- Evitando tests inestables
- Page Objects en Appium
- Corriendo Pruebas en SauceLabs

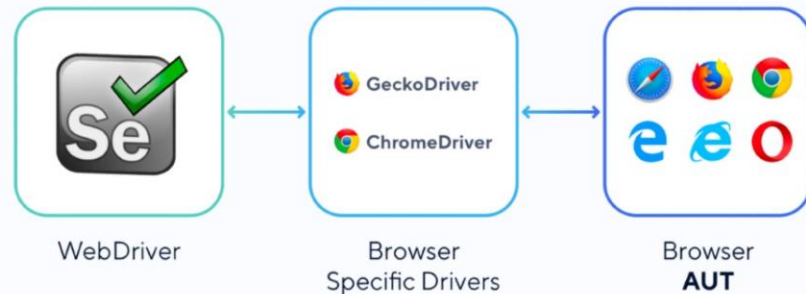


***HERRAMIENTAS QUE VAMOS A  
UTILIZAR.....***





## Selenium WebDriver Architecture







```

Given I prepare DB executing the script file "insert_MS02_Developers_Guide.sql" # com.chakray.q
  ngScriptFileSomething(java.lang.String)
And I set HTTP call endpoint to "enterpriseIntegrator" property with context "/library/bbdd/book/1" # com.chakray.q
  ltiSomethingPropertyWithContextString(java.lang.String,java.lang.String)
And I set the HTTP call header "Content-Type" with value "application/json" # com.chakray.q
  arWithValue(java.lang.String,java.lang.String)
When I make the HTTP call by GET method # com.chakray.q
  pCallBySomethingMethod(com.chakray.q,cucumber.support.type.CucumberParameterType$RequestMethod)
Then I check the HTTP status code received is 200 # com.chakray.q
  usCodeReceivedIs(int)
And the previous JSON content matches "MS02_Developers_Guide_response.json" # com.chakray.q
  eJsonContentWithFile(java.lang.String)

```

2 Scenarios (2 passed)  
13 Steps (13 passed)  
0m1.914s

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.025 sec - in com.chakray.RunCukeTest

Results :

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

```

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----

```



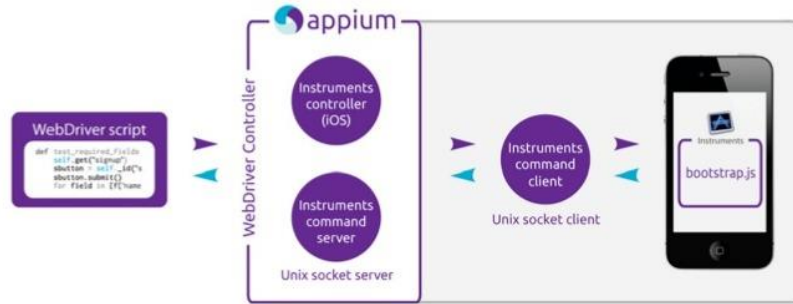
```

myDemoProject/pom.xml Test1.java
1 package myDemoProject_Test1;
2
3 import org.openqa.selenium.WebDriver;
4 import org.testng.annotations.Test;
5
6 public class Test1 {
7
8     WebDriver driver;
9
10    @Test
11    public void setUp() {
12        //your code here
13    }
14
15 }
16

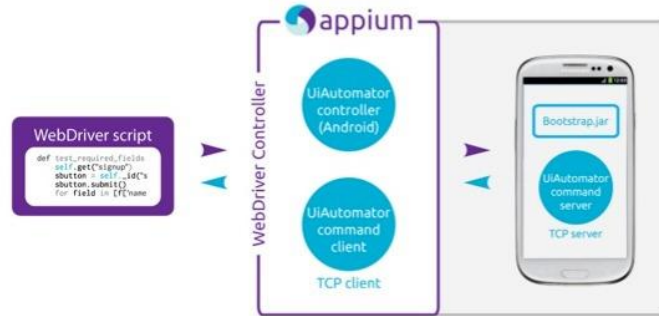
```



# IOS



# ANDROID

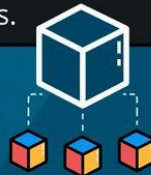


# ***PROGRAMACIÓN ORIENTADA A OBJETOS***



# ¿QUÉ ES LA PROGRAMACIÓN ORIENTADA A OBJETOS?

Es un **paradigma de programación** que organiza las funciones en entidades llamadas objetos.



- Los objetos se crean a partir de una **plantilla llamada clase**. Cada objeto es una instancia de su clase.

CLASE



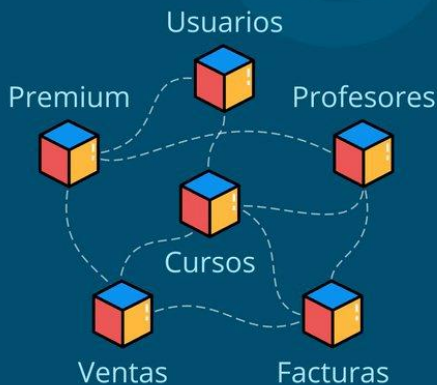
INSTANCIACIÓN

OBJETO



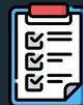
- Los objetos tienen **datos (atributos)** y **funcionalidades (métodos)**.

- En una aplicación los objetos están separados **pero se comunican entre ellos**.



ATRIBUTOS

Nombres  
Apellidos  
Correo  
Contraseña  
Premium

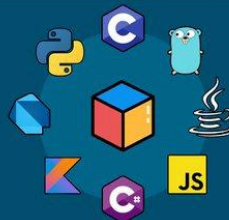


MÉTODOS

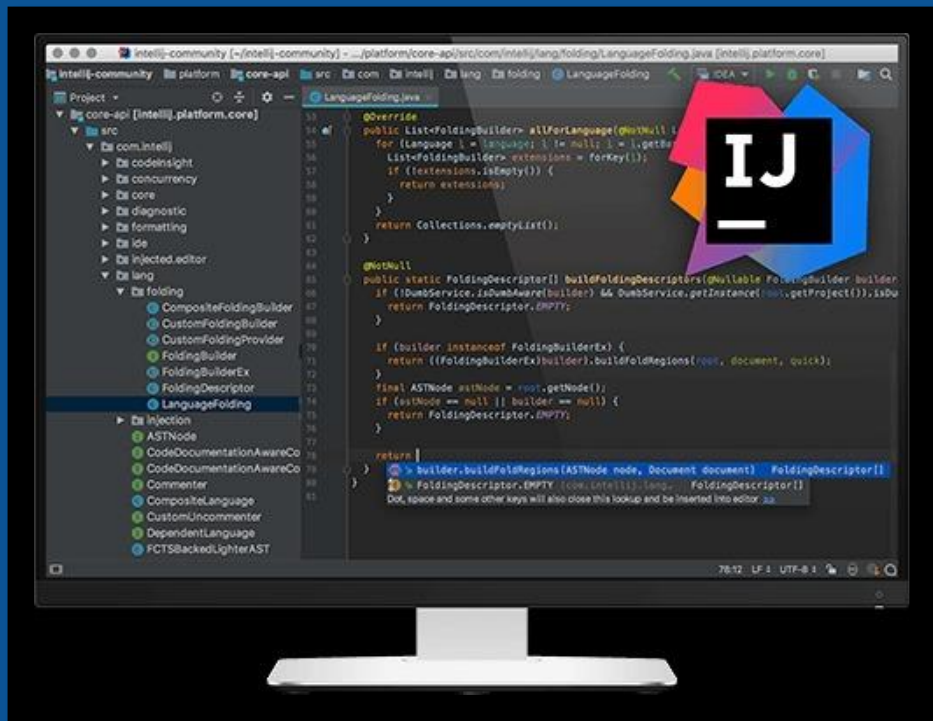
Editar perfil  
Iniciar sesión  
Cerrar sesión  
Cambiar contraseña  
Pasar a premium



Puedes programar con este paradigma **en la mayoría de lenguajes**.



# CREACIÓN DE UN PROYECTO JAVA Y PRIMERAS CLASES.





```
public class Nombre_Clase
{
    public static void main (String args[])
    {
        //bloque de sentencias;
    }
}
```

```
//Le damos un nombre "MiClase" a la clase
public class MiClase
{
    //Atributos de la clase
    private String atributo1;
    private int atributo 2;
    private float atributo 3;

    //Constructor con el mismo nombre de la clase
    public MiClase(){ }

    //Métodos de la clase
    public void metodo1()
    {
        //Método vacio
    }

    public String metodo2()
    {
        return "metodo2";
    }
}
```

```
public Animal(String nuevoNombre)
{
    nombre = nuevoNombre; //Se le da un nombre al animal
}


//Método para obtener la edad del animal
public int getEdad()
{
    return edad;
}

//Método para establecer la edad del animal
public void setEdad(int nuevaEdad)
{
    edad = nuevaEdad;
}

//Método para obtener el nombre del animal
public String getNombre()
{
    return nombre;
}
```







```
package ruta.del.paquete;
```

```
package otro_paquete.mi_paquete;  
/*Se usa el punto para separar cada carpeta  
   equivale a la ruta otro_paquete/mi_paquete dentro del proyecto*/  
public class mi_clase  
{  
  
}
```

```
package paquete.mipaquete;
```

```
import paquete2.otropaquete.*;
```

```
class Ejemplo1{}
```

```
package paquete.mipaquete;
```

```
import paquete3.otropaquete2.MiClase;
```

```
class Ejemplo2{}
```



# ¿Cómo empezamos...?

